Exploiting Social Property for Improving Distributed Semantic Search

Juan Li[a]
North Dakota State University, U.S.A

Abstract

To locate desirable Semantic Web data in a distributed network, the discovering mechanism has to be not only semantically rich, in order to cope with complex queries, but also scalable to handle large numbers of information sources. In this paper, we propose a novel scheme that exploits the social property of humans, such as natural grouping and peer recommendation between people with common interests, to expedite the discovery of Semantic Web data in large-scale distributed networks. In this framework, network nodes perform local dynamic topology adaptations to spontaneously create communities according to users' social-closeness. The basic premise of such semantic communities is that search requests have a high probability of being fulfilled within the community they originate from. For queries which cannot be efficiently solved inside the community, an index overlay built on Distributed Hash Table (DHT) is used to assist the search. Recommendations from peers with similar interests are employed to improve both the efficiency and the precision of the semantic search. Experiments with simulations substantiate that our techniques significantly improve the search efficiency, scalability, and precision.

Keywords: Semantic Web, search, overlay, query, social behavior.

1. Introduction

Semantic Web has been presented as an evolving extension of World Wide Web [1, 2, 3]. With the development of semantic web technologies, more and more semantic web data are generated and widely used in Web applications and enterprise information systems. These data are structured with ontologies [4] for the purpose of comprehensive and transportable machine understanding. To fully utilize the large amount of semantic data, an effective search mechanism customized for Semantic Web data, especially for ontologies, is needed by human users as well as software agents and services. The unique semantic features and the inherent distributed nature of Semantic Web data make its discovery highly challenging.

Peer-to-peer (P2P) technology has been used as a solution to distributed resource discovery, since it scales to very large networks, while ensuring high autonomy and fault-tolerance. The recently proposed structured P2P systems in the form of DHTs [5-8] are a promising approach for building massively distributed data management platforms. However, they offer few data management facilities, limited to Information Retrieval (IR) -style keyword search. Keyword search is appropriate for simple file-sharing applications, but is unable to deal with complex semantic queries which have various properties and sophisticated relations with each other.

More recently, a few studies [9, 10] extended the DHT-based P2P to support semantic queries. The basic idea is to map each keyword of a semantic entity to a key. For example, RDFPeer [9] indexes each RDF [20, 21] triple to support semantic RDF query. A query with multiple keywords then uses the DHT to lookup each keyword and returns the intersection. Systems like [8] avoid this multiple lookup and intersection by storing a complete keyword list of an object on each node. In this way, the DHTs can support multi-keywords queries. However, DHTs still have difficulty to support other richer queries, such as wildcard queries, fuzzy queries, and proximity queries. In addition, most DHT-based applications require all peers in the system sharing a uniform ontology schema, which is impractical in reality. These limitations restrict the deployment of DHTs to Semantic Web data discovery.

To support more complex queries, many P2P systems [11, 12] use flooding or maintain a broadcasting structure, such as a tree or a super cube, to propagate the queries to the network. For example, to execute an RDF query, Edutella [11] broadcasts the query to the whole hypercube. These approaches can support arbitrary types of queries. However, the overhead of flooding and broadcast may cause scalability and efficiency issues.

To overcome the shortcomings of existing discovery approaches, we propose a novel search mechanism which utilizes the social behavior of participating peers and integrates structured DHT P2P technology with unstructured P2P technology. In our system, each user/node is associated with a semantic summary representing the user's social interests. Based on the summary, we design a method to compute the semantic similarity between different users.

[a]Computer Science Department
Fargo, ND 58108, USA
j.li@ndsu.edu

The network topology is then reconfigured with respect to nodes' semantic similarity, so that peers with similar interests are close to each other, forming a semantic community. The semantic community is loosely structured as an unstructured P2P overlay, called community overlay. Because it has no structure requirements, the community overlay is able to handle flexible complex queries. The semantic locality property guarantees that the query evaluation can be limited to relevant peers only. A structured DHT-based overlay is used to facilitate the construction of the community overlay and to assist query evaluations that cannot be effectively resolved by the community overlay.

Members in the same community share similar interests hence are able to make recommendations to each other. A node can get useful information from other peers' recommendations. For example, before propagating a question to the network, a node can first check what other peers recommend for similar questions. Moreover, recommendation feedback from semantically similar peers can be employed to retrieve the most relevant results thus improving the efficiency and precision of searching. Recommendations not only help disambiguate search requests quickly, but also personalize query results for users by ranking higher the results that are relevant to users' semantic properties. Therefore, the search quality in terms of both precision and recall is improved. Finally, peers recommend "friends" for each other to adapt to the evolving network topology.

With the assistance of peer recommendation, community overlay and directory overlay complement each other, providing efficient search for the system. Compared to search in pure structured P2P systems, our hybrid search system has inherent support for complex semantic query or partial match; in addition, the retrieved results are more relevant. Compared to search in pure unstructured P2P systems, our community-based structure saves the overhead of flooding the query to unrelated nodes, thus enjoying more scalability.

The remainder of this paper is organized as follows. Section 2 gives an overview of the system framework. Section 3 describes one main component of the system – the community overlay. Section 4 presents another major component of the system – the index overlay. Section 5 introduces the query evaluation process. Section 6 explains how peer recommendations can be used to improve the discovery performance. In Section 7, we evaluate the proposed strategies a comprehensive set of simulations. Related work and concluding remarks are provided in Sections 8 and 9, respectively.

2. System Overview

In a human society, people naturally form all kinds of social networks by occupation, by interest, by location, etc. People always utilize these communities to gain knowledge or share information. In addition, people find that recommendations from friends (i.e., contacts of the social network) are more useful. The motivation behind the system design is based on these observations and the fact that the computer network is analogous to human social network. Both computer networks and human communities consist of members that are actively engaged in the sharing, communication and promotion of common interests. We believe we can exploit the social properties of the participants to improve sharing and discovering in the semantic network.

The proposed system consists of two logical overlays – an unstructured community overlay and a structured index overlay – taking different roles for efficient operations of the system. Query evaluation is mainly performed in the community overlay. In a community overlay, peers are connected to those sharing common interests; as a result, the propagation of a query tends to first reach those that are more likely to possess the data being searched for. This semantic locality property enables the community overlay to answer most queries originated from the local community. Unlike DHTs, community overlay does not specify any requirements for the query format, hence is able to handle any arbitrary types of complex queries. For the above reasons, a large portion of queries can be resolved inside the local community. However, peers are likely to have more diverse interests which cannot be covered by the community overlays they belong to. Therefore, there may exist a portion of queries that cannot be resolved by searching the community overlay alone. In this case, the index maintained by the index overlay can be consulted for hints about where to forward the query for a second try.

The index overlay is built on top of DHT protocols. It provides a high-level directory service for the system by indexing abstract ontology skeletons. The index overlay has two main functionalities: (1) It facilitates the construction of the community overlay. (2) It resolves unpopular queries or queries that are not covered by the community overlay. Unlike community overlay, index overlay does not give exact answers of a particular query; instead, it locates all peers possessing keywords of the query. Then the query will be broadcasted to all peers related to the keywords for further evaluation.

The distinction of community overlay and index overlay is virtual. A physical node may be involved in both of these two overlays. Community overlay and index overlay benefit from each other: index overlay facilitates the construction of the community overlay, while feedback from communities improves the search precision of the index overlay. Working together, these two overlays improve the efficiency and accuracy of the system.

Recommendations are used by both the community overlay and the index overlay to improve search performance. In the community overlay, peers share similar interests, therefore it is beneficial for them to recommend information to each other. Nodes in the community overlay maintain a cache to proactively exchange recommendation information. In the index overlay, an indexed keyword may have multiple meanings, not all of these meanings match the requestor's intention. Simply forwarding the query to all peers containing the keywords is not accurate and consumes lots of unnecessary network bandwidth. The index overlay employs peers' recommendation and feedback to solve the aforementioned semantic ambiguity problem. After receiving results from the index overlay, the requestor first checks the validity of the results. Then it reports its findings back to the index overlay nodes. The feedback will benefit future requesters with similar interests.

In the following sections, we introduce the construction and usage of these two overlays and how recommendation can be applied to improve the efficiency and precision of searching in these two overlays.

3. Community Overlay

The construction of the community is a topology adaptation process, i.e., to make the system's dynamic topology match the semantic clustering of peers. When a new node joins the network, existing nodes in the network recommend some neighbors to it according to their semantic similarity. A node also gradually updates its links according to recommendations of its neighbors and its query experiences, so that its topology always reflects its changing interests and data contents. The community topology enables queries to be quickly propagated among relevant peers. In addition, this topology allows semantically related nodes to establish ontology mappings.

3.1 Semantic similarity

There has been extensive research [17 - 19] focusing on measuring the semantic similarity between two objects in the field of information retrieval and information integration. However, their methods are very comprehensive and computationally intensive. In this paper, we propose a light-weight method to compute the semantic similarity between two nodes.

Our system supports semantic web data represented as OWL ontology. OWL ontology can be divided into two parts: the terminological box (TBox) and the assertion box (ABox) as defined in the description logic terminology [16]. TBox ontology defines the high-level concepts and their relationships. It is a good abstraction of the ontology's semantics and structure. Therefore, we use a node's TBox ontology to represent its semantic interest. In particular, we use keywords of a node's TBox ontology as its ontology summary. However, a semantic meaning may be represented by different keywords in different ontologies, while it is also possible that the same keyword in different ontologies means totally different things. Ontology comparison based on TBox keywords may not yield satisfying results. In order to solve this problem, we extend each concept with its semantic meanings in WordNet [22]. We use two most important relationships in WordNet – synonyms and hypernym – to expand concepts. In this way, semantically related concepts would have overlaps.

After extension, a node's ontology summary set may get a number of unrelated words, because each concept may have many senses (meanings), but not all of them are related to the ontology context. A problem causing the ambiguity of concepts is that the extension does not make use of any relations in the ontology, which are important clues to infer the semantic meanings of concepts. To further refine the semantic meaning of a particular concept, we utilize relations between the concepts in an ontology to remove unrelated senses from the summary set. Since the dominant semantic relation in an ontology is the *subsumption* relation, we use the *subsumption* relation and the sense disambiguation information provided by WordNet to refine the summary. It is based on a principle that a concept's semantic meaning should be consistent with its super-class's meaning. We use this principle to remove those inconsistent meanings. For every concept in an ontology, we check each of its senses; if a sense's hypernym has overlap with this concept's parent's senses, then we keep this sense and the overlapped parent's sense to the ontology summary set. Otherwise, they are removed from the set. In this way we can refine the summary and reduce imprecision.

To compare two ontologies, we define an ontology similarity function based on the refined ontology summary. The definition is based on Tversky's "Ratio Model" [23] which is evaluated by set operations and is in agreement with an information-theoretic definition of similarity [24]. Assume A and B are two nodes, and their ontology summary are S(A) and S(B) respectively. The semantic similarity between node A and node B is defined as:

$$sim(A,B) = \frac{|S(A) \cap S(B)|}{|S(A) \cap S(B)| + \alpha\,|S(A) - S(B)| + \beta\,|S(B) - S(A)|}$$

In the above equations, "∩" denotes set intersection, "–" is set difference, while "| |" represents set cardinality, "α" and "β" are parameters that provide for differences in focus on the different components. The similarity *sim*, between A and B, is defined in terms of the semantic concepts common to A and B: S(A)∩S(B), the concepts that are distinctive to A: S(A)–S(B), and the features that are distinctive to B: S(B) – S(A). Two nodes, node A and node B are said to be semantically related if their semantic similarity measure, sim(A,B) exceeds the user-defined similarity threshold t ($0<t\leq 1$).

3.2 Community construction

The construction of an ontology-based overlay is a process of finding semantically related neighbors. A node joins the network by connecting to one or more bootstrapping neighbors. The bootstrapping neighbors try to recommend some other neighbors to this new node according to its semantic property. If the bootstrapping neighbors do not have such recommendation information at hand, they will issue a neighbor-discovery query for the new node. The neighbor discovery query contains the new joining node's ontology summary in the form of a Bloom Filter [25]. The bootstrapping neighbors then use strategies (such as [13-15]) to efficiently propagate the neighbor discovery query to the network. Nodes receiving the query compute their semantic similarity with the new node based on the semantic summary. Semantically related nodes then return positive replies to the new node. If there are not enough neighbors discovered within the hops limited by TTL, the new node will turn to the index overlay for assistance. After the neighbor-discovery process, a new node is positioned to the right community. Inside the community overlay, nodes randomly connect with their neighbors. Queries looking for particular contents can be forwarded inside the community overlay using flooding- or random-walk- based simple forwarding algorithms.

Because of the dynamic property of the large-scale network, and the evolution of nodes' ontology property, neighbor discovery for a node is not once and for all, but rather the first-step of the topology adaptation scheme. A node should keep updating its neighbor links according to its query experiences and other peer's recommendations. A peer obtains experiences by accumulating queries received as a query forwarding router, and query results collected as a query requestor. Personal experiences can be forwarded to friends/peers as recommendations. Based on its experiences and peer recommendations, a node may add or delete neighbors according to the dynamic semantic environment. This way, the network topology is reconfigured with respect to peers' semantic properties, and peers with similar ontologies are close to each other.

4. Index Overlay

As a facilitator and complement of the community overlay, the index overlay indexes top-level semantic interests and unpopular semantic concepts. The popularity of a concept is determined according to the query history. As mentioned, OWL ontology can be divided into two parts: TBox and ABox. Similar to a database schema, a node's TBox knowledge is more abstract, describing the node's high-level concepts and their relationships. In contrast, ABox includes concrete data and relations, for example, the instances of classes defined in the TBox. Index overlay indexes TBox and ABox ontology for different purpose: TBox indexing helps nodes locate communities while ABox indexing assists nodes finding instances which cannot be quickly located in the community overlay. Only unpopular ABox instances are indexed in the index overlay.

The index overlay is constructed according to the mechanism of the corresponding structured P2P overlay. We employ RDFPeer's indexing method presented by M. Cai *et al* [9]. The basic idea is to divide RDF description into triples and then index the triples in a DHT overlay. We store each triple three times by applying a hash function to its *subject*, *predicate*, and *object*. In this way, a query providing partial information of a triple can be handled. For TBox indexing, the three parts of a triple may be uneven: a TBox has only a limited number of predefined predicates, but many more objects and subjects. For example, many classes have a *subClass* property; each is encoded as a triple with predicate *rdf:subClass*. When indexing by the predicate, all these triples are mapped to the same key and therefore to the same peer of the index overlay. This causes overloading of the peer in charge of the key. This problem can be solved by simply not indexing the overly popular keys; the query can be resolved by using other information of the triple.

Peers register their top semantic interests in the form of TBox ontology through the *insert(key,value)* operation in the index overlay. The index overlay node in charging of that key maintains a Least Recently Used (LRU) cache storing contact information of registered peers. A neighbor discovery query can get contacts of other peers interested in the same ontology through this index overlay node. Then the new node can connect with these contacts and join their community. At the same time, the new node registers to the index overlay by adding itself to the cache of the indexing node. A node with multiple interests can register with multiple indexing nodes. The index overlay also indexes unpopular ABox instances which cannot be quickly located inside the community.

5. Semantic Query Evaluation

With the hybrid topology constructed, semantic query can be efficiently evaluated. Owing to the semantic locality, in most cases, a resource discovery query can be answered within the querying node's local community overlay. The semantic community reduces the search time and decreases the network traffic by minimizing the number of messages circulating between nodes. There are many strategies, such as [13-15] to effectively propagate queries in an unstructured P2P network. Popular data items are more likely to be located quickly since they have more replicas in the community, whereas an unpopular data item cannot be found unless a large number or all of the peers are searched. Also, queries for data in other semantic communities are unlikely to be solved inside the local community overlay. For these cases, nodes turn to the index overlay to get assistance.

Index overlay indexes top semantic interests and unpopular instances, thus is able to give hints to queries which cannot be solved by the community overlay. For example, a node can find interested community by lookup the interest in the index overlay, then contact those nodes returned by the index overlay. A node can also find unpopular ABox instances from index overlay which may not easily be found from the community overlay.

6. Peer Recommendation

Peer recommendation can further improve the efficiency of query evaluation, and at the same time, improves the query accuracy. The recommendation is performed in two ways with difference functionalities: The first scheme is push-based (or proactive) recommendation, in which peers proactively recommend data or friends to others with similar interests. The other recommendation scheme is pull-based (or retroactive) in which peers get refined query results based on previous recommendations from similar peers. The former improves the query efficiency and reduces query latency, while the latter enhances the accuracy of the query results.

To realize the proactive recommendation, each peer maintains an active recommendation cache storing queries and corresponding results. Socially close peers (with semantic similarity beyond a threshold *t*) exchange their cached data. Because these peers are similar, one peer's query will probably be queried by another peer in the future. These cached data work as good recommendations for peers' future query. These recommendations behave like a cache with high hit-rate. They will greatly improve the query efficiency by intercepting peers' queries, so that the queries will not be propagated to the network. Besides queries, peers also recommend friends to each other. The same as in social network, in a semantic community, similar peers should share common friends.

The retroactive recommendation is mainly used in the DHT index overlay. For ABox instances, DHT indexing has the semantic ambiguity problem. For example, it is difficult to figure out whether the search term *palm* is a company (company: palm), a technology (operating system: palm), or a product (PDA:palm). We solve the ambiguity problem with community feedback recommendations. To facilitate query refinement with the community feedback, the indexing peers need to perform some additional tasks: besides storing the ABox keywords, an indexing peer is also responsible for maintaining clusters of peers related to each sense of the keyword.

Figure 1 shows an example of a data entry stored in an indexing peer. There are six peers related to the term, *palm*. Initially when a node issues a query related to term *palm* trying to find information about a PDA, all six peers are returned to the requester as shown in Figure 1 (a). The requester will contact each of them, although only three of them (P_1, P_{13}, P_2) are related to PDA. After the requester contacts all these six peers and evaluates their data, it returns its feedback (i.e., which peers have the data with right senses) to the indexing peer. The indexing peer will link those three related peers with the requester's community, as shown in Figure 1 (b). Next time, a requester from the same community will take advantage of this clustering and be given only the three related peers. In this way, the precision of the query evaluation is improved and the network traffic is reduced.

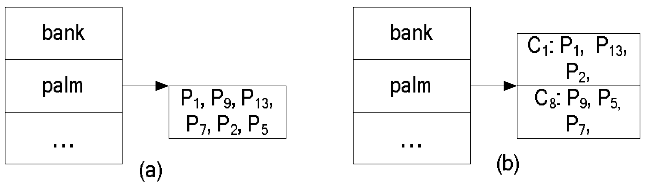

Figure 1. Example of a data entry stored in an index node

7. Experiment

Owing to the lack of access to the semantic environment with many nodes, our system performance evaluation falls back to simulations. We first describe the experimental setup, and then analyze the simulation results.

7.1 Experimental setup

As it is difficult to find representative real world ontology data, we have chosen to generate test data artificially. Our data does not claim to model real data, but shall rather provide reasonable approximation to evaluate the performance of the system. Ontology data can be characterized by many factors such as the number of classes, properties, and individuals; thus we have generated the test data in multiple steps. The algorithm starts with generating the ontology schema (TBox). Each schema includes the definition of a number of classes and properties. The classes and properties may form a multilevel hierarchy. Then the classes are instantiated by creating a number of individuals of the classes. To generate an RDF instance triple *t*, we first randomly choose an instance of a class C among the classes to be the subject: *sub(t)*. A property *p* of C is chosen as the predicate *pre(t)*, and a value from the range of *p* to be the object: *obj(t)*. If the range of the selected property *p* are instances of a class C', then *obj(t)* is a resource; otherwise, it is a literal.

The queries are generated by randomly replacing parts of the created triples with variables. For our experiments, we use single-triple-queries and conjunctive-triple-queries. To create the conjunctive-queries, we randomly choose a property p_1 of class C_1. Property p_1 leads us to a class C_2 which is the range of p_1. Then we randomly choose a property p_2 of class C_2. This procedure is repeated until the range or the property is a literal value or we have created n ($n \leq 3$) triple patterns.

In our experiments, the total number of distinguished ontologies is 100. We assume each node uses 1 to 3 ontologies. Each ontology includes at most 10 classes. The number of properties that each class has is at most k=3. The number of instances of each class at each peer is less than 10. Finally, the number of triple patterns in each query we create is either 1 or 3. In our experiment, we do not do knowledge reasoning. In other words, we do not augment the RDF graph by inference (forward chaining). Other simulation parameters and their default values are listed in Table 1.

The simulation is initialized by injecting nodes one by one into the network until a certain network size has been reached. The network topology created this way has power-law properties; nodes inserted earlier have more links than those inserted later. This property is consistent with the real world situation, in which nodes with longer session time have more neighbors. After the initial topology is created, a mixture of joins, leaves, and queries are injected into the network based on certain ratios. The proportion of join to leave operations is kept the same to maintain the network at approximately the same size. Inserted nodes start functioning without any prior knowledge.

Table 1. Parameters used in the simulations

Parameter	**Range and default value**
network size	2^9~2^{15} default: 10,000
initial neighbors (node degree)	5
maximum neighbors	30
average node degree	14
TTL	1~20 default 9
ontology domains	100
ontology schemas per domain	1~10 default:8
distinct resources per domain	100
resources per node	1~10
die/leave probability per time slice per node	0-21%, 3% default
resource change probability per time slice per node	20%instance update, 2% schema update
query probability per time slice per node	5%
sample of nodes to compute diameter	5%

The index overlay is implemented as a Pastry [6] virtual network in Java. Each peer is assigned a 160-bit identifier, representing 80 digits (each digit uses 2 bits) with base b=2. After the network topology has been established, nodes publish their TBox knowledge and some unpopular ABox data on the overlay network. Then nodes are randomly picked to issue queries. Each experiment is run ten times with different random seeds, and the results are the average of these ten sets of results.

7.2 Results

In this part, we present the experimental results which demonstrate the performance of our searching scheme.

7.2.1 Emergence of the social community

As discussed, the topology of the peer network is a crucial factor determining the efficiency of the search system. We expect that our semantics-based topology formation scheme will transform the topology into a social community. To verify this transformation, we examine two network statistics, the *clustering coefficient* and the *average network path length*, as indicators of how closely the topology has approached a “small-world” [26] topology.

The *clustering coefficient* (*CC*) is a measure of how well connected a node’s neighbors are with each other. According to one commonly used formula for computing the *clustering coefficient* of a graph (Eq. 1), the *clustering coefficient* of a node is the ratio of the number of existing edges and the maximum number of possible edges connecting its neighbors. The average over all |V| nodes gives the *cluttering coefficient* of a graph (Eq. 2).

$$CCv = \frac{\text{\# of edges between v's neighbors}}{\text{maximum \# of possible edges between v's neighbors}} \quad (1)$$

$$CC = \frac{1}{|V|}\sum_{v} CCv \quad (2)$$

The *average path length (APL)* is defined as the average shortest path across all pairs of nodes (Eq. 3). The *APL* corresponds to the degree of separation between peers. For a large graph, measuring distances between all node pairs is computationally expensive; therefore an accepted procedure is to measure it over a random sample of nodes [27]. In our experiment, we use a random sample of certain percent of the graph nodes. We use Dijkstra’s algorithm to compute the shortest distance between pairs of nodes. In our simulated topology we intentionally make the network strongly connected, so that any pairs of nodes have a directed path.

$$APL = \frac{\sum_{ij} l_{i,j}}{|V| \cdot (|V|-1)} \quad (3)$$

We performed experiments to measure the *cluster coefficient* (*CC*) and *average path length (APL)* of our proposed system. An interest-based ShortCut topology and a random power-law topology with the same average node degree are used as reference topologies. The former has been proved to be a small-world system [28]. For the ShortCut scheme, test results are collected after the system has had an extensive training process, i.e., nodes have learned as many ShortCuts as possible through query results and the system topology has become stable.

Figures 2 and Figure 3 show plots of the *clustering coefficient* and the *average path length* as a function of the number of nodes in the network. We observe that both the *clustering coefficient* and the *average path length* of our semantic clustering are very similar to those of ShortCut. The *clustering coefficients* of semantic-clustering and ShortCut are much larger than that of the random power-law network, while the *average path length* of semantic clustering and ShortCut are almost the same as that of the random network. This indicates the emergence of a small-world network topology [27]. Note: Because all of the three topologies are created by inserting nodes to the existing system, all topologies show the power-law property to some extent, and thus the *average path length* of all three topologies are smaller than a random network. This set of experiments verifies that firstly, well connected clusters exist in the proposed system; due to the semantic similarity definition, these clusters correspond to groups of users with shared ontological interests. Secondly, there is, on average, a short path between any two nodes in the system topology graph; therefore, queries with relatively small TTL would cover most of the network.

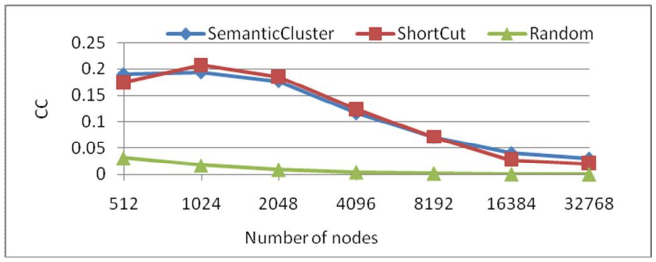

Figure 2. Comparison of clustering coefficient

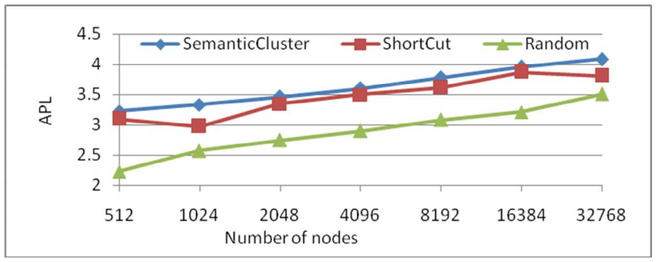

Figure 3. Comparison of average path length

7.2.2 Efficiency of the hybrid topology

We expect the community overlay and the index overlay work together to dramatically improve the system performance. To verify this conception, we examine the system performance in two different aspects, namely scalability and efficiency by executing the experiment in different network configurations. The performance is measured using an Information Retrieval (IR) standard: *recall*. Recall refers to completeness of retrieval of relevant items, as defined below:

$$recall = \frac{|relevantDocuments \cap retrievedDocuments|}{|relevantDocuments|}$$

First, we vary the number of nodes from 2^9 to 2^{15} to test the scalability of the system. The results are listed in Figure 4. Our hybrid system gets higher recall in all these different sized networks. In addition, our recall decreases less with the increase in network size.

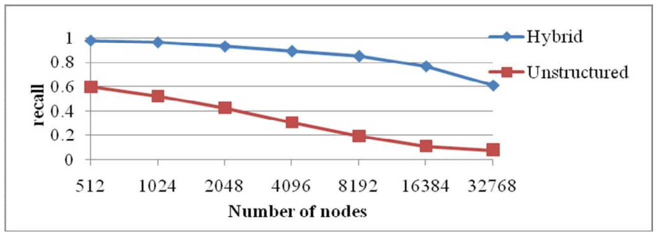

Figure 4. Recall rate vs. network size

Figure 5 illustrates the system efficiency by showing the relationship between query recall rate and query TTL. With a small TTL, our system gets a higher recall rate, i.e., resolves queries faster.

As expected, the hybrid topology based on the two overlays performs well as measured by recall rate. The semantic community topology effectively reduces the search space, and its ontology summary indexed by the index overlay guides the query in the right direction. Therefore, it can locate results faster and more accurately. This explains why the proposed topology scales to large network size and why it achieves higher recall with shorter TTL.

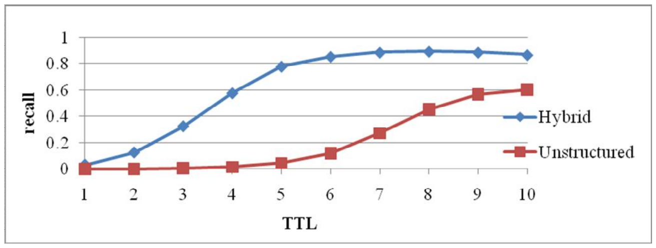

Figure 5. Recall rate vs. TTL

7.2.3 Effect of recommendation

We distinguish the two types of recommendation: the proactive and the retroactive recommendation and examine their effect respectively. To testify the effect of proactive recommendation, we compare the scenario where peers proactively exchange their recommendation cache with that where peers do not exchange recommendation at all. We show that our recommendation cache improves the query performance by reducing not only query traffic but also query latency. Figure 6 and Figure 7 demonstrate these two aspects respectively. In this experiment, we increase the skew degree of the query distribution from random to Zipf [29] with α=1.25. From Figure 6 and Figure 7 we can get two conclusions: (1) proactive recommendation significantly reduces the query traffic and query latency. (2) Query distribution has a significant impact on the performance of recommendation. The more skewed the query distribution, the more effective the recommendation performs. According to [30], in an open and live distributed environment, query distribution is skewed and follows a Zipf distribution. Therefore, our recommendation scheme would be an effective strategy to improve the system performance.

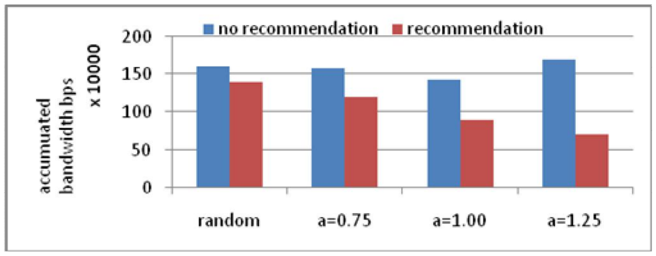

Figure 6. Performance of proactive recommendation (accumulated bandwidth vs. query distribution)

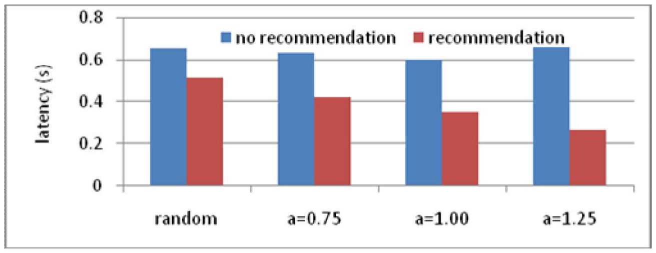

Figure 7. Performance of proactive recommendation (query latency vs. query distribution)

To testify the effect of retroactive recommendation, we create a special experimental scenario which uses a small-sized dictionary D to generate the ontology data. We randomly pick S words from D, representing polysemy or homonymy (words with multiple meanings); if these words appear in different communities, they represent different meanings. In this experiment, we count the number of nodes visited to find 30 results at different time period. As shown in Figure 8, with the time going, using community feedback may reduce the number of nodes to be explored. Because feedback from communities helps eliminating semantic ambiguity of the index overlay, queries are only forwarded to the most relevant nodes. Consequently, the precision of the search is increased.

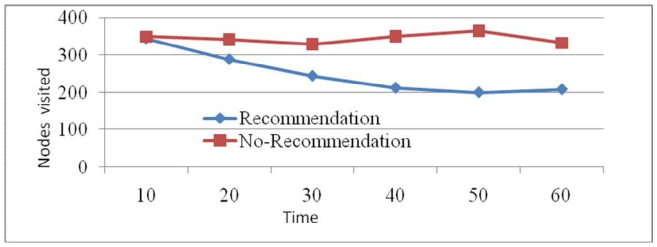

Figure 8. Effect of retroactive recommendation

8. Related work

Most current research on searching or querying Semantic Web uses an Information Retrieval (IR)-based central search engine (e.g. [31–35]). The IR-based work, such as Swoogle [32] and SWSE [33], indexes the Semantic Web by crawling and indexing the Semantic Web RDF documents found online and then offers a search interface over these documents. However, the centralized server can be a potential bottleneck when the number of users is large. In addition, the IR-based semantic data search does not provide structured query capability.

To address the scalability issue, researchers have utilized P2P technologies to Semantic Web. For example, systems such as Edutella [11] and InfoQuilt [36] use broadcast or flooding to search RDF data, while many other projects, like RDFPeer [9] and OntoGrid [10] attempt applying DHT techniques to the retrieval of the ontology encoded knowledge. The flooding- based approaches create too much network traffic while the DHT-based approaches cannot support complex queries.

Recently, there has appeared the idea of grouping nodes with similar contents together to facilitate search [39, 28, 38, 39]. The latest super-peer-based Edutella [38] and Semantic Overlay Network (SON) [40] rely on centralized server or super-peers to cluster contents and nodes. Preliminary work in [28] proposes to cluster nodes with similar interest together, without discussing how to define the interest similarity amongst peers and how to form clusters. [39] relies on periodic message exchanges amongst peers to keep track of other peers with similar documents, which incurs very high message overhead. Semantic Small Word (SSW) position peers and data objects in the semantic space, so that peers with similar data objects form into clusters. It then applies a dimension reduction technique on top of the DHT to realize a semantics-based search. In SSW, semantics of data objects is represented by a multi-attribute vector, but not Semantic Web-based data. Applications such as REMINDIN [41], Helios [42], and Bibster [43] add semantic short-cuts to group nodes. The short-cut approach relies on the presence of interest-based locality. Each peer builds a shortcut list of nodes that answered previous queries. To find content, a peer first queries the nodes on its shortcut list and only if unsuccessful, floods the query.

9. Conclusions

This paper presents an effective framework for semantic query evaluation in a large-scale distributed network. Our system exploits the social network property to improve the efficiency and accuracy of query evaluation. It combines the structured and unstructured P2P topology to form a hybrid topology that includes two types of overlays: community overlay and index overlay. Community overlay is formed according to nodes' semantic similarity, so that queries can be focused in semantically related regions only. For queries that cannot be effectively resolved in the semantic community, they can be solved by the index overlay. Recommendations from peers are used to predict future queries and disambiguate the meanings of the queries. Simulation experiments demonstrate that this framework improves the scalability, efficiency, and precision of search in a large semantic heterogeneous network.

References

[1] Tim Berners-Lee (with Mark Fischetti), "Weaving the Web, The original design and ultimate destiny of the World Wide Web", Harper, 1999.

[2] T. Berners-Lee, J. Hendler, and O. Lassila, "The Semantic Web: A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities," Scientific American, May 2001.

[3] D. Fensel and M. Musen, Eds. "The Semantic Web: A Brain for Humankind," IEEE Intelligent Systems, March/April 2001.

[4] T. R. Gruber, "Principles for the Design of Ontologies Used for Knowledge Sharing." International Journal Human-Computer Studies, 43(5-6):907-928, 1995.

[5] B. Y. Zhao, J. D. Kubiatowicz, and A. D. Joseph. "Tapestry: An Infrastructure for Fault-Tolerant Wide-Area Location and Routing," Technical Report, UCB 2000.

[6] A. Rowstron and P. Druschel. "Pastry: Scalable, Distributed Object Location and Routing for Large-Scale Peer-to-Peer Systems," in Proc. of the IFIP/ACM CDSP, Middleware, 2001.

[7] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H.Balakrishnan. "Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications," ACM SIGCOMM, 2001

[8] S. Ratnasamy, P.Francis, M.Handley, R.Karp, and S. Shenker. "A Scalable Content-Addressable Network," ACM SIGCOMM, 2001.

[9] M. Cai, M. Frank, "RDFPeers: A scalable distributed RDF repository based on a structured peer-to-peer network", in proc of WWW conference, NewYork, USA, May 2004.

[10] OntoGrid project: http://www.ontogrid.net/

[11] W. Nejdl et al. "EDUTELLA: a P2P Networking Infrastructure Based on RDF". In Proceedings of the 11th international conference on World Wide Web (WWW), 2002.

[12] M. Arumugam, A. Sheth, and I. B. Arpinar. "Towards peer-to-peer semantic web: A distribuited environment for sharing semantic knowledge on the web." In Proc. of the International World Wide Web Conference 2002 (WWW2002), Honolulu, Hawaii, USA, 2002.

[13] Juan Li and Son Vuong, "SOON: A Scalable Self-Organized Overlay Network for Distributed Information Retrieval", in Proceedings of the 19th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management Managing Large Scale Service Deployment 2008.

[14] Y. Chawathe, S. Ratnasam, L. Breslau, N. Lanhan, S. Shenker, "Making Gnutella-like P2P Systems Scalable", In Proceedings of ACM SIGCOMM'03, 2003.

[15] Juan Li, Son Vuong, "Efa: an Efficient Content Routing Algorithm in Large Peer-to-Peer Overlay Networks", in Proceedings of the Third IEEE International Conference on Peer-to-Peer Computing", 2003.

[16] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, P. F. Patel-Schneider: The Description Logic Handbook: Theory, Implementation, Applications. Cambridge University Press, Cambridge, UK, 2003. ISBN 0-521-78176-0

[17] J. Jiang and D. Conrath, "Semantic Similarity Based on Corpus Statistics and Lexical Taxonomy," in Proceeding of the Int'l Conf. Computational Linguistics, 1997.

[18] J. Lee, M. Kim, and Y. Lee, "Information Retrieval Based on Conceptual Distance in IS-A Hierarchies," J. Documentation, vol. 49, pp. 188-207, 1993.

[19] M. A. Rodriguez, M. J. Egenhofer, "Determining Semantic Similarity Among Entity Classes from Different Ontologies". IEEE Transactions on Knowledge and Data Engineering, 2003.

[20] O. Lassila and Ralph R. Swick, "W3C Resource Description framework (RDF) Model and

Syntax Specification", World Wide Web Consortium, 1999.

[21] Dan Brickley and R.V.Guha. "W3C Resource Description Framework (RDF) Schema Specification". http://www.w3.org/TR/1998/WD-rdf-schema/

[22] C. Fellbaum. "WordNet: An Electronic Lexical Database," In Fellbaum, Christiane, MIT Press, 1998.

[23] A. Tversky. Features of similarity. Psychological Review, 84(4):327–352, 1977.

[24] D. Lin. An information-theoretic definition of similarity. In Proc. 15th International Conf. on Machine Learning, pages 296–304. Morgan Kaufmann, San Francisco, CA, 1998.

[25] B. Bloom. "Space/time tradeoffs in hash coding with allowable errors". Communications of the ACM, pages 13(7):422-426, July 1970.

[26] A. L. Barabási, "Linked: How Everything is Connected to Everything Else and What It Means for Business, Science, and Everyday Life." New York, Plume, 2003.

[27] D. Watts and S. Strogatz. "Collective dynamics of "small-world" networks." - Nature, 1998.

[28] A. Iamnitchi, M. Ripeanu, and I. Foster. Small-world filesharing communities. In Proceedings of the Infocom, Hong Kong, China, 2004.

[29] G. K. Zipf, "Human Behaviour and the Principle of Least-Effort", Addison-Wesley, Cambridge MA, 1949.

[30] L. Adamic, B. Huberman, "Zipf's law and the Internet" Glottometrics, 2002.

[31] Zhang, L., Liu, Q., Zhang, J., Wang, H., Pan, Y., Yu, Y.: Semplore: An IR approach to scalable hybrid query of semantic web data. In: Proceedings of the 6th International Semantic Web Conference, 2007.

[32] Ding, L., Finin, T., Joshi, A., Pan, R., Cost, R.S., Peng, Y., Reddivari, P., Doshi,V., Sachs, J.: Swoogle: a search and metadata engine for the semantic web. In: Proc. of the 13th ACM CIKM Conf. (2004).

[33] Hogan, A., Harth, A., Umbrich, J., and Decker, S. "Towards a scalable search and query engine for the web". In Proceedings of the WWW 2007.

[34] Guha, R., McCool, R., Miller, E.: Semantic search. In: Proc. of the 12th Intl. Conf. on World Wide Web. (2003).

[35] Rocha, C., Schwabe, D., Aragao, M.P.: A hybrid approach for searching in the semantic web. In: Proc. of the 13th Intl. Conf. on World Wide Web. (2004).

[36] M. Arumugam, A. Sheth, and I. B. Arpinar. "Towards peer-to-peer semantic web: A distribuited environment for sharing semantic knowledge on the web." In Proc. of the International World Wide Web Conference 2002 (WWW2002), Honolulu, Hawaii, USA, 2002.

[37] M. Bawa, G. S. Manku, and P. Raghavan. SETS: Search enhanced by topic segmentation. In Proceedings of ACM SIGIR, pages 306–313, July 2003.

[38] W. Nejdl, M. Wolpers, W. Siberski, C. Schmitz, M. T. Schlosser, I. Brunkhorst, and A. Lser. Super-peer-based routing and clustering strategies for RDF-based peer-to-peer networks. In Proceedings of International World Wide Web Conference (WWW), pages 536–543, May 2003.

[39] C. H. Ng, K. C. Sia, and C. H. Chang. Advanced peer clustering and firework query model in the peer-to-peer network. In Proceedings of International World Wide Web Conference (WWW), May 2003.

[40] A. Crespo and H. Garcia-Molina. Semantic overlay networks. Technical report, Stanford University, 2002.

[41] X. Tempich, S. Staab, A. Wranik, "REMINDIN': semantic query routing in peer-to-peer networks based on social metaphors" International World Wide Web Conference (WWW), New York, USA, 2004.

[42] A. Castano, S. Ferrara, Montanelli, and D. Zucchelli. Helios: a general framework for ontology-based knowledge sharing and evolution in P2P systems. In IEEE Proc. of DEXA WEBS 2003 Workshop, Prague, Czech Republic, September 2003.

[43] A. Castano, S. Ferrara, S. Montanelli, E. Pagani, G. Rossi, : Ontology addressable contents in p2p networks. In: Proceedings of the WWW'03 Workshop on Semantics in Peer-to-Peer and Grid Computing, 2003.

Juan Li is an assistant professor in the department of Computer Science at North Dakota State University. She received her Ph.D. degree in computer science from the University of British. Dr. Li's research focus is distributed systems including Peer-to-Peer systems, mobile ad hoc networks, grid computing, and semantic web technologies. She has conducted extensive research in distributed query management, semantics-based search, and P2P network, and published many conference papers, journal papers, and book chapters in these areas. Dr. Li has given presentations at national and international conferences and workshops. She is an active member of the Association for Computing Machinery (ACM) and the IEEE Computer Society.